Deduction as Reduction

Dominique Duval
LJK, University of Grenoble
Dominique.Duval@imag.fr

October 30., 2010

Abstract. Deduction systems and graph rewriting systems are compared within a common categorical framework. This leads to an improved deduction method in diagrammatic logics.

1 Introduction

Deduction systems and graph rewriting systems can be seen as different kinds of reduction systems. In this article, they are compared within a common categorical framework. This leads to an improved deduction method in diagrammatic logics.

On the one hand, category theory may be used for describing graph rewriting systems: this is called the algebraic approach to graph rewriting [Corradini et al. 1997, Ehrig et al. 1997]. On the other hand, proofs may be modelled by morphisms in a category [Lambek 1968, Lawvere 1969]. However, these are two disjoint research topics. In this paper, we present a common categorical framework for dealing with graph rewriting and with logical deduction.

Our deduction systems are defined using *diagrammatic logics* [Duval 2003, Domínguez and Duval 2010], where a logic is a functor $\mathcal{L} : \mathbf{S} \to \mathbf{T}$ satisfying several properties. In this framework, a major role is played by *pleomorphisms* (called *entailments* in previous papers): for a given logic $\mathcal{L} : \mathbf{S} \to \mathbf{T}$, a pleomorphism is "half-way" between a general morphism and an isomorphism; it is a (generally non-reversible) morphism in the category $\mathbf{S}$ which is mapped by the functor $\mathcal{L}$ to a reversible morphism in the category $\mathbf{T}$. In biology, a pleomorphism is the occurrence of several structural forms during the life cycle of a plant; in diagrammatic logic, a pleomorphism refers to the occurrence of several presentations of a given logical theory during a proof: various lemmas are progressively added to the given axioms until the required theorem is obtained. In this paper, the analogy with graph rewriting systems extends this approach in such a way that it becomes possible to drop intermediate lemmas.

Section 2 is devoted to graph rewriting systems and section 3 to deduction systems, then the comparison is done in section 4. Some familiary is assumed from the reader with the notions of categories, functors and pushouts. It is recalled that a span (resp. a cospan) in a category is simply a pair of morphisms with the same domain (resp. codomain).

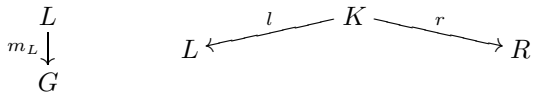
2 Reduction: graph rewriting

Well-known examples of *reduction systems* (or *rewriting systems*) are *string* rewriting systems, *term* rewriting systems and *graph* rewriting systems. Let us focus on the last ones. A graph rewriting system consists of a binary relation $\rightsquigarrow$ on graphs, i.e., a set of *rewrite rules* of the form $L \rightsquigarrow R$. Given a rewrite rule $L \rightsquigarrow R$ and an occurrence (called a *match*) of L in a graph G, the *rewrite step* consists of "replacing" the occurrence of L in G by an instance of R, which gives rise to a new graph H. This can be applied to various families of graphs, and the notion of "replacement" may take various meanings.

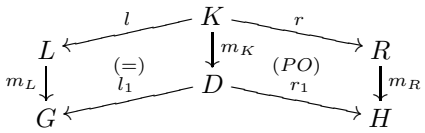
In the algebraic approach, graph rewriting systems are described in a categorical framework. Such

systems include the *double pushout* (DPO), *simple pushout* (SPO), *sesqui-pushout* (SqPO) and *heterogeneous pushout* (HPO) approaches [Ehrig et al. 1997, Corradini et al. 1997, Corradini et al. 2006, Duval et al. 2009].

In this paper we focus on the double pushout and the sesqui-pushout graph rewriting systems. Given a category $\mathbf{C}$ of graphs, in both approaches a *match* is a morphism $m_L : L \longrightarrow G$ in $\mathbf{C}$ and a *rewrite rule* $L \rightsquigarrow R$ is a span (l, r) between L and R in $\mathbf{C}$



Let us call *generalized pushout under a span* (l, r) a diagram of the following form, with a commutative square on the left and a pushout square on the right



In both graph rewriting systems the *rewrite step* builds a generalized pushout. First m_K and l_1 are built from m_L and l so as to get a commutative square, then m_R and r_1 are built from m_K and r so as to get a pushout. In both approaches there are restrictions on the form of the matches and rules.

- In the double pushout approach, the left square is a pushout, which means that the construction of m_K and l_1 from m_L and l is a *pushout complement*.
- In the sesqui-pushout approach, the left square is a pullback, and more precisely the construction of m_K and l_1 from m_L and l is a *final pullback complement*.

3 Deduction: diagrammatic logic

Deduction systems, in this paper, are defined in the framework of *diagrammatic logic* [Duval 2003, Domínguez and Duval 2010]. We only need to know that a diagrammatic logic is a pushout-preserving functor $\mathcal{L} : \mathbf{S} \to \mathbf{T}$ between two categories with pushouts. Then $\mathbf{T}$ is called the category of *theories* of the logic $\mathcal{L}$ and $\mathbf{S}$ its category of *specifications*. Each specification Σ *presents*, or *generates*, the theory $\mathcal{L}\Sigma$.

In order to get a full definition one must add that $\mathcal{L}$ is the left adjoint in an adjunction induced by a morphism of limit sketches [Ehresmann 1968] and that it makes $\mathbf{T}$ a category of fractions over $\mathbf{S}$ [Gabriel and Zisman 1967]. This full definition, which will not be used in this paper, enlightens the importance of pleomorphisms (definition 3.1) in diagrammatic logic: in fact, in this situation the pleomorphisms determine the functor $\mathcal{L}$ [Gabriel and Zisman 1967].

Definition 3.1. With respect to some given diagrammatic logic $\mathcal{L} : \mathbf{S} \to \mathbf{T}$:

- Two specifications Σ, Σ' in $\mathbf{S}$ are *pleoequivalent* if there is an isomorphism of theories $\mathcal{L}\Sigma \cong \mathcal{L}\Sigma'$; this is denoted $\Sigma \underline{\underline{\ }} \Sigma'$.
- An *instance* of Σ_1 in Σ_2, where Σ_1 and Σ_2 are specifications, is a morphism $\sigma' : \Sigma_1 \longrightarrow \Sigma_2'$ in $\mathbf{S}$ where Σ_2' is pleoequivalent to Σ_2; this is denoted $\Sigma_1 \longrightarrow \Sigma_2' \underline{\underline{\ }} \Sigma_2$.
- A *pleomorphism* is a morphism of specifications $\tau : \Sigma \longrightarrow \Sigma'$ such that $\mathcal{L}\tau$ is an isomorphism of theories; this is denoted $\tau : \Sigma \underset{\leftarrow}{\rightarrow} \Sigma'$.
- A *fraction* from Σ_1 to Σ_2 is a cospan $(\sigma : \Sigma_1 \longrightarrow \Sigma_2', \tau : \Sigma_2 \underset{\leftarrow}{\rightarrow} \Sigma_2')$ in $\mathbf{S}$ where τ is a pleomorphism; this is denoted $\tau \backslash \sigma : \Sigma_1 \longrightarrow \Sigma_2' \underset{\rightarrow}{\leftarrow} \Sigma_2$. The *numerator* of $\tau \backslash \sigma$ is σ, its *denominator* is τ and its *vertex* is Σ_2'.

Remark 3.2. Clearly, when two specifications are related by a zig-zag of pleomorphisms, they are pleoequivalent: the equivalence relation generated by the pleomorphisms is included in the pleoequivalence relation.

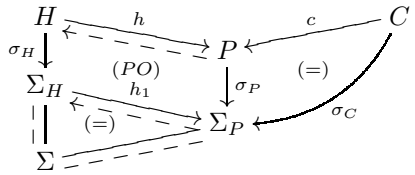
The next result states some straightforward properties of pleomorphisms.

Proposition 3.3. *Pleomorphisms satisfy the following properties:*

- *every isomorphism in* $\mathbf{S}$ *is a pleomorphism,*
- *if* $h = g \circ f$ *in* $\mathbf{S}$ *and if two among* f, g, h *are pleomorphisms, then so is the third,*
- *pleomorphisms are stable under pushouts.*

Definition 3.4. Given a diagrammatic logic $\mathcal{L} : \mathbf{S} \to \mathbf{T}$

- A *deduction rule* (or *inference rule*) is a fraction $h\backslash c : C \longrightarrow P \xleftarrow[=]{} H$ from C to H. The *hypothesis* of $h\backslash c$ is H, its *conclusion* is C.
- The *deduction step* with respect to a deduction rule $h\backslash c : C \longrightarrow P \xleftarrow[=]{} H$ maps each instance $\sigma_H : H \longrightarrow \Sigma_H \underline{=} \Sigma$ of H in some specification Σ to the instance $\sigma_C : C \longrightarrow \Sigma_C \underline{=} \Sigma$ of C in the same Σ defined as follows (where h_1 is a pleomorphism because so is h and pleomorphisms are stable under pushouts)



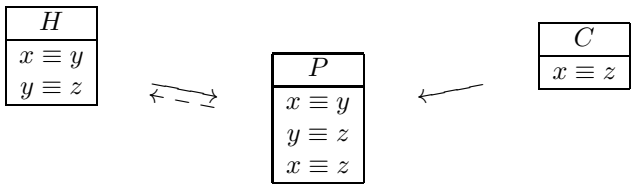
Remark 3.5. A deduction rule $h\backslash c$ has numerator c and denominator h, in contrast with the usual notation $\frac{H}{C}$. Indeed, it is the morphism h, and not c, which becomes an isomorphism of theories.

Remark 3.6. This construction is essentially the composition of fractions in their bicategory.

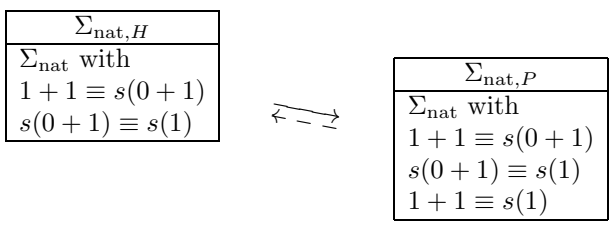
Example 3.7. Let $\mathcal{L}_{\text{eq}}$ be the equational logic. One of its rule is the *transitivity* rule

$$\frac{x \equiv y \quad y \equiv z}{x \equiv z}$$

which corresponds to the fraction



Let Σ_{nat} be the equational specification "of naturals" made of a sort N, a constant $0 : N$, two operations $s : N \to N$, $+ : N^2 \to N$ and two equations $0 + y \equiv y$, $s(x) + y \equiv s(x + y)$. Let us analyze the last step in the proof of $1 + 1 \equiv s(1)$ (where 1 stands for $s(0)$), once it has been proved that $1 + 1 \equiv s(0 + 1)$ and that $s(0 + 1) \equiv s(1)$, so that it remains to use the transitivity rule in order to conclude. Then $\Sigma_{\text{nat},H}$ is Σ_{nat} together with the terms $1 + 1$, $s(0 + 1)$, $s(1)$ and the equations $1 + 1 \equiv s(0 + 1)$, $s(0 + 1) \equiv s(1)$. The deduction step yields $\Sigma_{\text{nat},P}$, made of $\Sigma_{\text{nat},H}$ together with the equation $1 + 1 \equiv s(1)$, and σ_C maps $x \equiv z$ to $1 + 1 \equiv s(1)$, as required.



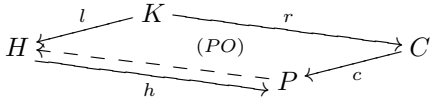
4 Deduction as Reduction

It is clear from the previous sections that deduction is a form of reduction, as well as graph rewriting.

- In a graph rewriting system (section 2), given a rewrite rule $L \rightsquigarrow R$ and a match of L in G, the rewrite step consists of "replacing" the occurrence of L in a graph G by an instance of R, which gives rise to a new graph H.
- In a deduction system (section 3), given a deduction rule $\frac{H}{C}$ and an instance of H in Σ with vertex Σ_H, the deduction step consists of "replacing" the occurrence of H in Σ by an instance of C, which gives rise to an instance of C in Σ with a new vertex Σ_C.

But a graph rewrite rule is a span while a deduction rule is a cospan, so that the descriptions of the rewrite steps are quite different. However, in this section, under the assumption that a deduction rule can also be defined from a span, we exhibit similarities between both reduction systems and we propose improvements in the construction of the deduction steps.

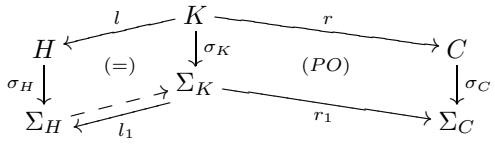
Assumption 4.1. It is now assumed that each deduction rule $h \backslash c : C \longrightarrow P \xleftarrow{} H$ is obtained from a pushout



Remark 4.2. Usually a deduction rule is given as $\frac{H}{C}$, neither P nor K are mentioned. Then K can be defined as the family of features which have the same name in H and in C, and P can be obtained from a pushout as in assumption 4.1.

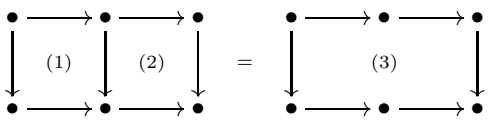
In section 2 we have seen double pushouts and sesqui-pushouts as instances of generalized pushouts. Now we define a third family of generalized pushouts.

Definition 4.3. Given a span (l, r), a *pleopushout under* (l, r) is a diagram of the following form, with a commutative square on the left, a pushout square on the right, where l_1 is a pleomorphism

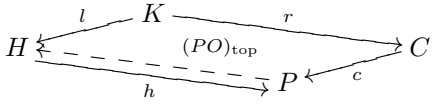


Theorem 4.5 below builds a deduction step from a pleopushout. In its proof we use the properties of pleomorphisms stated in proposition 3.3 and well-known properties of pushouts stated now in proposition 4.4.

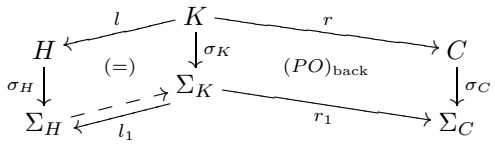
Proposition 4.4. *Given two consecutive commutative squares* (1), (2) *and the composed commutative squares* (3)*, if* (1) *is a pushout then* (2) *is a pushout if and only if* (3) *is a pushout.*



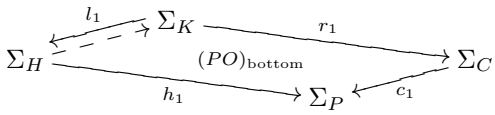
Theorem 4.5. *Let $h \backslash c$ be a deduction rule satisfying assumption 4.1*



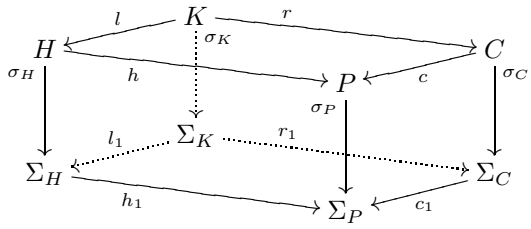
Let σ_H be an instance of H in Σ, and let us assume that there is a pleopushout under (l, r)



Let us consider the pushout



Then there is a unique morphism $\sigma_P : P \longrightarrow \Sigma_P$ such that we get a commutative cube

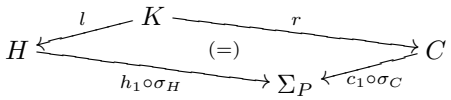


Then in this cube:

- *the top, bottom, back right and front left faces are pushouts,*
- *the morphism h and the four morphisms of the bottom face are pleomorphisms.*

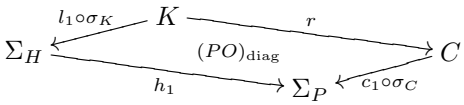
If σ_H is an instance of H in some Σ then σ_C is an instance of C in Σ.

Proof. It is easily checked that the following square is commutative

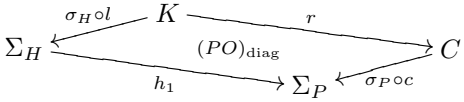


So, the pushout $(PO)_{\text{top}}$ gives rise to a unique morphism $\sigma_P : P \longrightarrow \Sigma_P$ such that the two front faces in the cube are commutative. Since the other faces are yet known to be commutative, the cube is commutative.

It is yet known that the top $(PO)_{\text{top}}$, bottom $(PO)_{\text{bottom}}$ and back right $(PO)_{\text{back}}$ faces are pushouts. Let us prove that the front left face is also a pushout. According to proposition 4.4, composing $(PO)_{\text{back}}$ and $(PO)_{\text{bottom}}$ gives rise to the "diagonal" pushout $(PO)_{\text{diag}}$



Thanks to the commutativity of the cube, the pushout $(PO)_{\text{diag}}$ can also be written as

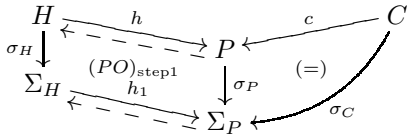


Now, since $(PO)_{\text{top}}$ and $(PO)_{\text{diag}}$ are pushouts, proposition 4.4 implies that the front left face of the cube is also a pushout $(PO)_{\text{front}}$.

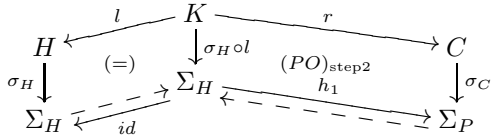
It is yet known that h and l_1 are pleomorphisms. Let us check that the three other morphisms of the bottom face are pleomorphisms, using proposition 3.3. Since pleomorphisms are stable under pushouts, it follows from $(PO)_{\text{bottom}}$ that c_1 is a pleomorphism and from $(PO)_{\text{front}}$ that h_1 is a pleomorphism. Since three among the four morphisms in the bottom commutative square are pleomorphisms, so is the fourth: hence r_1 is also a pleomorphism.

It follows that Σ_H and Σ_C are pleoequivalent, which proves the last assertion of the theorem. □

Remark 4.6. According to definition 3.4, the deduction step with respect to $h\backslash c$ is defined from the following diagram



If $h\backslash c$ satisfies assumption 4.1, then by proposition 4.4 the composition of $(PO)_{\text{top}}$ and $(PO)_{\text{step1}}$ yields a pushout $(PO)_{\text{step2}}$, which obviously forms the right part of a pleopushout under (l, r)

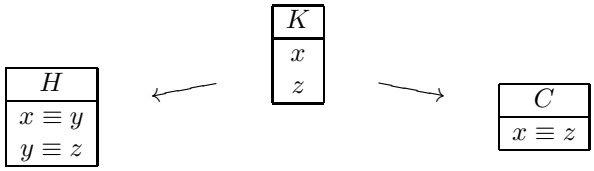


So, a pleopushout under (l, r) as assumed in theorem 4.5 is obtained from the deduction step (definition 3.4). This proves that indeed a deduction step can be seen as a reduction step. Moreover, theorem 4.5 states that whenever we are able to find a "better" pleopushout than this obvious one, then we may get an instance of C in Σ "better" than $\sigma_C : C \longrightarrow \Sigma_P$ in definition 3.4. Such a situation occurs in example 4.7.

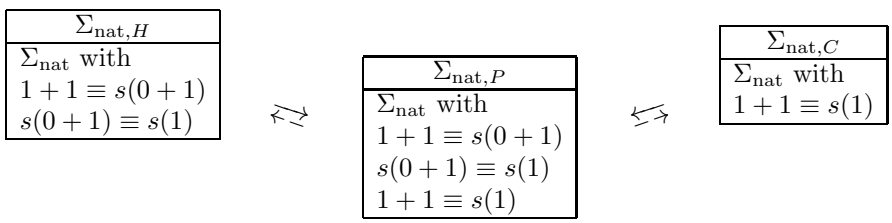
Example 4.7. As in example 3.7, let $\mathcal{L}_{\text{eq}}$ be the equational logic. The *transitivity* rule:

$$\frac{x \equiv y \quad y \equiv z}{x \equiv z}$$

can be obtained by a pushout from a span $H \longleftarrow K \longrightarrow C$:



As in example 3.7, let Σ_{nat} be the equational specification "of naturals" and let us analyze the last step in the proof of $1 + 1 \equiv s(1)$: it has yet been proved that $1 + 1 \equiv s(0 + 1)$ and $s(0 + 1) \equiv s(1)$, and it remains to use the transitivity rule. As in example 3.7, $\Sigma_{\text{nat},H}$ is Σ_{nat} together with the terms $1 + 1$, $s(0 + 1)$, $s(1)$ and the equations $1 + 1 \equiv s(0 + 1)$, $s(0 + 1) \equiv s(1)$. Let us define $\Sigma_{\text{nat},K}$ as Σ_{nat} with the terms $1 + 1$ and $s(1)$ (with no additional equations), with $\sigma_{\text{nat},K}$ which maps x to $1 + 1$ and z to $s(1)$ and with l_1 the inclusion. Then by pushout $\Sigma_{\text{nat},C}$ is made of Σ_{nat} with the equation $1 + 1 \equiv s(1)$. It is smaller than $\Sigma_{\text{nat},P}$ from example 3.7: the lemmas $1 + 1 \equiv s(0 + 1)$ and $s(0 + 1) \equiv s(1)$, which have been used during the proof, are kept in $\Sigma_{\text{nat},P}$ while they are dropped from $\Sigma_{\text{nat},C}$.



5 Conclusion

Deduction systems as well as graph rewriting systems can be seen as reduction systems. This paper lays the foudations for such comparisons. Further developments might involve adhesive categories [Lack and Sobocinski 2005]. This should provide a new point of view about the role of pullbacks in graph rewriting, as well as new methods for deduction in diagrammatic logics.

References

[Corradini et al. 1997] Andrea Corradini, Ugo Montanari, Francesca Rossi, Hartmut Ehrig, Reiko Heckel, Michael Lwe. Algebraic Approaches to Graph Transformation - Part I: Basic Concepts and Double Pushout Approach. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 1: Foundations*, World Scientific, p. 163–246 (1997).

[Corradini et al. 2006] Andrea Corradini, Tobias Heindel, Frank Hermann, Barbara Knig. Sesqui-Pushout Rewriting. ICGT'06. *Lecture Notes in Computer Science* 4178, Springer, p. 30–45 (2006).

[Domínguez and Duval 2010] César Domínguez, Dominique Duval. Diagrammatic logic applied to a parameterization process. *Mathematical Structures in Computer Science* 20(04) p. 639–654 (2010).

[Dumas et al. 2010] Jean-Guillaume Dumas, Dominique Duval, Jean-Claude Reynaud. Cartesian effect categories are Freyd-categories. *Journal of Symbolic Computation* (2010).

[Duval 2003] Dominique Duval Diagrammatic specifications. *Mathematical Structures in Computer Science* 13 p. 857–890 (2003).

[Duval et al. 2009] Dominique Duval, Rachid Echahed, Frédéric Prost. A Heterogeneous Pushout Approach to Term-Graph Transformation. RTA'09. *Lecture Notes in Computer Science* 5595, Springer, p. 194–208 (2009).

[Ehresmann 1968] Charles Ehresmann. Esquisses et types de structures algébriques. *Bull. Instit. Polit. Iaşi* XIV (1968).

[Ehrig et al. 1997] Hartmut Ehrig, Reiko Heckel, Martin Korff, Michael Lwe, Leila Ribeiro, Annika Wagner, Andrea Corradini. Algebraic Approaches to Graph Transformation - Part II: Single Pushout Approach and Comparison with Double Pushout Approach. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 1: Foundations*, World Scientific, p. 247–312 (1997).

[Gabriel and Zisman 1967] Peter Gabriel and Michel Zisman. *Calculus of Fractions and Homotopy Theory*, Springer (1967).

[Lack and Sobocinski 2005] Stephen Lack, Pawel Sobocinski. Adhesive and quasiadhesive categories. *Informatique Théorique et Applications* 39(3) p. 511–545 (2005).

[Lambek 1968] Joachim Lambek. Deductive systems and categories I. *Mathematical Systems Theory* 2(4) p. 287–318 (1968).

[Lawvere 1969] F. W. Lawvere. Adjointness in foundations. *Dialectica* 23 p. 281–296 (1969).